AD-A282 270

Improving Productivity in the Development of Large Software Systems

Final Report
Contract N00014-85-C-0710

DTIC
ELECTE
JUL 22 1994
S F

Submitted to

Advanced Research Projects Agency
Contracts Management (CMO)
3701 North Fairfax Drive.
Arlington, VA 22203-1714

This document has been approved for public release and sale; its distribution is unlimited

DTIC QUALITY INSPECTED 2

by

Software Options, Inc.
22 Hilliard Street
Cambridge, Mass. 02138

DTIC QUALITY INSPECTED 1

February 17, 1994

94-22741

94 7 19 219

The project began with a re-examination of the roles and boundaries of programming languages and environments. The belief was that present technology was a result of historical drift rather than technical foresight. The expectation was that a fresh look at modern hardware and programming practices would result in significantly better languages and environments. By "better" we mean the results would, for example, reduce redundant effort and increase automation of implicit tasks, in order to allow the programmer to focus on the content of a problem. These are vague and lofty goals but our plan was specific and concrete. We developed appropriate formal underpinnings on which we based the implementation of infrastructures for developing languages and environments. We used the infrastructures to design and implement a new language and environment, called E-L.

We addressed language issues [Kard] that had not been satisfactorily addressed in any existing language, such as module interface specifications, extensible types, type templates and type resolution, overloading, and the relationship among state, equality and conversion [Kar87b]. We also embraced important language features that were not novel, such as functions and types as first class values, functions returning multiple values, programs as data, and syntactic extension. In addition to specific language issues, we developed an infrastructure for designing and implementing languages. It was clear then, as it is now, that (1) one language will never suit all purposes; (2) program analysis tools are important and costly to implement; and (3) the number of different machine architectures will continue to increase. Given these factors, we developed a formalism for defining a language [Kar88, KH91] and implementing a translator [KTb, Kar94] for it to a small, simple kernel. Tools operate at the kernel-level, including interpreters [Kar87a], analyzers [KRb, KRc], and optimizers. From this small, target-independent kernel, code generators produce target-specific instructions [Mor91]. We used the kernel-based approach to implement the E-L language, and it is being used at Harvard to study language design and program transformation issues for high performance architectures. The kernel-based approach is also playing a role in the design of a debugger interface.

This approach to language design and implementation creates an open "compiler" with many steps and tools involved in going from what a programmer writes to optimized code on a variety of targets. The consequence increases the advantages of an environment with better configuration and version management than RCS- and make-based tools provide. With this context in mind and a desire to support literate programming [CHK] and teams of programmers working cooperatively on a network of geographically distributed workstations [Kari], we implemented a development environment, called the Artifacts System [Karb, Towa]. This too is an open infrastructure. Though it presents a uniform and seamless interface to the user, it is extensible at various levels: through the integration of editors, tools [KHRc], and repositories. The purpose of the Artifacts System is to structure complex, evolving data, to assist users in their cooperative effort to develop such data, and to integrate the tools that operate on and produce this data. A key element in the design is to eliminate what is the usual interaction with a computer-based system: run a tool to achieve a desired effect. Rather, users of the Artifacts System set up structures that indicate desired results and browse these structures in hypertext-like fashion; tool invocation is usually implicit. Moreover, version and configuration management is an integral part of the system, not a facility on the side. The Artifacts System has been in daily use at Software Options for several years and hosts its own development. It currently supports programming [Towb] in C [Karh] and Common Lisp [KMH] and is awaiting the completion of GNAT to include Ada. The next step is to broaden the community of users, perhaps by finding a partner to commercialize it or by finding a small amount of funding to make it suitable for access and use via anonymous *ftp*.

We described the third major result of the project, a code generator based on a novel approach to code generation called coagulation [Kar84, Mor91], in earlier reports. We have recently begun a new ARPA-supported project to adapt the coagulation ideas to high performance computers.

We conclude by listing the many reports produced in the final phase of this contract, all of which are available from Software Options. We are also including copies of five documents [Karb, Towa, KHRc, Kar94, KR89] that are relatively comprehensive descriptions of the work.

Accesion For	
NTIS CRA&I	✓
DTIC TAB	☐
Unannounced	☐
Justification	
By [illegible]	
Distribution /	
Availability Codes	
Dist	Avail and / or Special
A-1	

References

[CHK] Thomas E. Cheatham, Glenn Holloway, and Michael Karr. *A Literate Implementation in the Artifacts System of the Game of Life.* Software Options, Inc., 22 Hilliard Street, Cambridge MA 02138.

[E-L] *E-L System.* Software Options, Inc., 22 Hilliard Street, Cambridge, MA 02138. Internal implementation documents.

[HCLK] Glenn H. Holloway, Thomas E. Cheatham, Russell Lopez, and Michael Karr. Artifacts system C utilities. E-L implementation document in [E-L].

[HK] Glenn H. Holloway and Michael Karr. Notices and pending plexes. E-L implementation document in [E-L].

[HKZ] Glenn Holloway, Michael Karr, and Alex Zatsman. Shellscript artifacts. E-L implementation document in [E-L].

[Kara] Michael Karr. Artist. E-L implementation document in [E-L].

[Karb] Michael Karr. *Beyond the Read-Eval Loop: The Artifacts System.* Software Options, Inc., 22 Hilliard Street, Cambridge MA 02138.

[Karc] Michael Karr. Drafts. E-L implementation document in [E-L].

[Kard] Michael Karr. *E-L Design Rationale.* Software Options, Inc., 22 Hilliard Street, Cambridge MA 02138.

[Kare] Michael Karr. E-L terms. E-L implementation document in [E-L].

[Karf] Michael Karr. Extender's guide for language implementations. E-L implementation document in [E-L].

[Karg] Michael Karr. Extender's guide for the E-L system. E-L implementation document in [E-L].

[Karh] Michael Karr. New C artifacts implementation. E-L implementation document in [E-L].

[Kari] Michael Karr. Plexes. E-L implementation document in [E-L].

[Kar84] Michael Karr. Code generation by coagulation. In *Proceedings of the ACM SIGPLAN '84 Symposium on Compiler Construction*, pages 1–12, June 1984. (SIGPLAN Notices (19,6)).

[Kar87a] Michael Karr. E-L 68000 runtime system. Technical Report SOI-02-87, Software Options, Inc., March 1987.

[Kar87b] Michael Karr. Equality, state and logic in the design of E-L. Technical Report SOI-14-87, Software Options, Inc., March 1987.

[Kar87c] Michael Karr. Very long executions. Technical Report SOI-19-87, Software Options, Inc., July 1987.

[Kar88] Michael Karr. Wreaths. Technical Report SOI-02-88, Software Options, Inc., July 1988.

[Kar94] Michael Karr. *E-L Definition.* Software Options, Inc., 22 Hilliard Street, Cambridge MA 02138, February 1994.

[KH91] Michael Karr and Paul Hudak. *A Kernel Family and the Common Prototyping System.* Software Options, Inc., 22 Hilliard Street, Cambridge MA 02138, January 1991.

[KHL] Michael Karr, Glenn Holloway, and Russ Lopez. Porting E-L. E-L implementation document in [E-L].

[KHMR] Mike Karr, Glenn Holloway, Chip Morris, and Steve Rozen. Generic implementation of artifact support for languages. E-L implementation document in [E-L].

[KHRa] Michael Karr, Glenn Holloway, and Steve Rozen. Derivative classes. E-L implementation document in [E-L].

[KHRb] Michael Karr, Glenn Holloway, and Steve Rozen. EMACS utilities. E-L implementation document in [E-L].

[KHRc] Michael Karr, Glenn Holloway, and Steve Rozen. Extender's guide for artifacts and drafts. E-L implementation document in [E-L].

[KHRL] Michael Karr, Glenn Holloway, Steve Rozen, and Russ Lopez. Artifacts. E-L implementation document in [E-L].

[KLM$^+$] Michael Karr, Russ Lopez, Chip Morris, Steve Rozen, and Glenn Holloway. LaTeX artifacts. E-L implementation document in [E-L].

[KMH] Michael Karr, Chip Morris, and Glenn Holloway. Common Lisp artifacts. E-L implementation document in [E-L].

[KRa] Michael Karr and Steve Rozen. Derivatives in the E-L system:implementation. E-L implementation document in [E-L].

[KRb] Michael Karr and Steve Rozen. Dynamic cross reference model implementation. E-L implementation document in [E-L].

[KRc] Michael Karr and Steve Rozen. Flow-graph model: Implementation. E-L implementation document in [E-L].

[KR89] Michael Karr and Steve Rozen. Models and interpretations: Formalizing multiple semantics. Extended abstract. Technical Report SOI-07-89, Software Options, Inc., December 1989.

[KTa] Michael Karr and Judy Townley. Modal E-L implementation. E-L implementation document in [E-L].

[KTb] Michael Karr and Judy G. Townley. Lattices and wreaths. E-L implementation document in [E-L].

[LK] Russ Lopez and Michael Karr. The E-L locking mechanism. E-L implementation document in [E-L].

[LKHC] Russell Lopez, Michael Karr, Glenn H. Holloway, and Thomas E. Cheatham. E-L servers. E-L implementation document in [E-L].

[Lop] Russ Lopez. The remote procedure call mechanism. E-L implementation document in [E-L].

[Mor91] Walter G. Morris. CCG: A prototype coagulating code generator. In *ACM SIGPLAN '91 Conference on Programming Language Design and Implementation*, pages 45–58, June 1991.

[MZSKa] Robert Morris, Alex Zatsman, Chris Small, and Michael Karr. Alarm-clock server. E-L implementation document in [E-L].

[MZSKb] Robert Morris, Alex Zatsman, Chris Small, and Michael Karr. Very long execution mechanism. E-L implementation document in [E-L].

[RLK] Steve Rozen, Russ Lopez, and Michael Karr. Distributed execution in the E-L system: Design and implementation. E-L implementation document in [E-L].

[RM] Steve Rozen and Walter G. Morris. Worker kernels (with implementations). E-L implementation document in [E-L].

[Towa] Judy G. Townley. *E-L Users' Manual.* Software Options, Inc., 22 Hilliard Street, Cambridge MA 02138.

[Towb] Judy G. Townley. *Language Users' Manual.* Software Options, Inc., 22 Hilliard Street, Cambridge MA 02138.